greatlearning
Power Ahead

Customer Segmentation using various Clustering Algorithms

Objectives

greatlearning

- **Context:** To perform segmentation using Clustering Techniques for a given dataset of customers visiting a mall

- Four techniques will be applied and compared viz. KMeans, DBSCAN, MeanShift and Agglomerative

- There will be some further discussions about algorithms alterations and their current developments/research

Clustering

greatlearning

- Clustering belongs to unsupervised machine learning techniques.
- The main task of clustering is to **discover "natural" groups in an unlabelled dataset**. This is an important task in data analysis as it is **used in many scientific, engineering and business applications**. The **most well-known application of clustering are customers segmentation (for efficient marketing), image segmentation, documents clusterisation**. There are many clustering algorithms which can be divided into two main types: hierarchical and partitional.
- **Hierarchical** algorithms **recursively split a dataset into a smaller subset** until a subset contains only one item. This can be represented with a dendrogram which looks like a tree. It can be constructed from leaves to the root (agglomerative approach) or from the root down to the leaves (divisive approach). In hierarchical clustering, you don't have to specify the number of clusters but you have to define a termination condition for splitting/merging process.
- **Partitional** algorithms **divide a dataset into several subsets (clusters) based on a given criteria**. For some algorithms number of clusters has to be defined a priori (e.g K-Means) and for some not (DBSCAN). **Defining the number of clusters** before running an algorithm often **requires a specific domain knowledge which is often challenging (or even impossible) in many applications**. This led to **the development of many heuristics and simplified approaches helping analyst without domain knowledge to choose the appropriate number of clusters**.

Clustering Techniques

greatlearning

- There is a vast number of clustering algorithms and currently, there is no single one that dominates other ones. Choosing the best one depends on the dataset itself, an application domain and client requirements and expectations
  - **K-means**
  - **DBSCAN**
  - **Hierarchical clustering**
  - **Mean Shift**
  - **Affinity Propagation**
  - **Spectral clustering**
  - **OPTICS**
  - **BIRCH**

- All are implemented in a well-known Python library: Scikit-Learn.

Comparison

greatlearning

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters, inductive	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry, inductive	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with `n_samples`	Many clusters, uneven cluster size, non-flat geometry, inductive	Distances between points
Spectral clustering	number of clusters	Medium `n_samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry, transductive	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters or distance threshold	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, transductive	Distances between points
Agglomerative clustering	number of clusters or distance threshold, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances, transductive	Any pairwise distance
DBSCAN	neighborhood size	Very large `n_samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes, transductive	Distances between nearest points
OPTICS	minimum cluster membership	Very large `n_samples`, large `n_clusters`	Non-flat geometry, uneven cluster sizes, variable cluster density, transductive	Distances between points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation, inductive	Mahalanobis distances to centers
BIRCH	branching factor, threshold, optional global clusterer.	Large `n_clusters` and `n_samples`	Large dataset, outlier removal, data reduction, inductive	Euclidean distance between points

Points to ponder

greatlearning

- Non-flat geometry clustering is useful when the clusters have a **specific shape**, i.e. a non-flat manifold, and the standard euclidean distance is not the right metric.
- Gaussian mixture models, useful for clustering, are dedicated to mixture models. KMeans can be seen as a special case of Gaussian mixture model with **equal covariance** per component.
- Transductive clustering methods (in contrast to inductive clustering methods) are **not designed to be applied to new, unseen data**.

KMeans

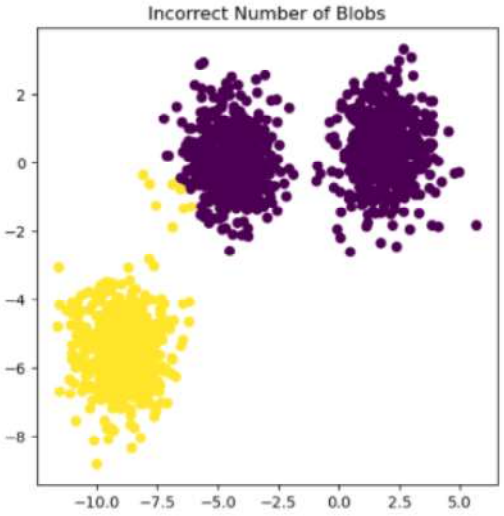
greatlearning

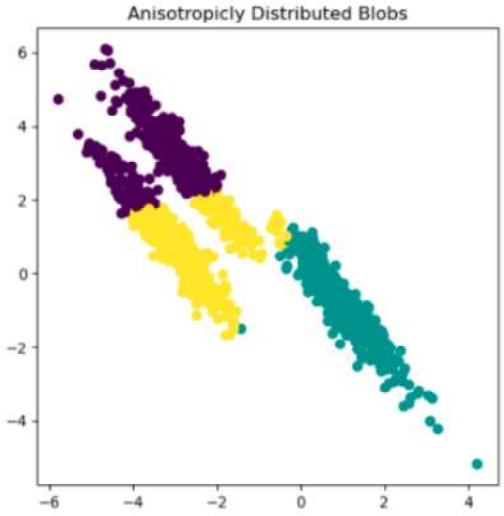
- The **KMeans** algorithm clusters data by trying to separate samples in n-groups of equal variance, **minimizing a criterion known as the *inertia* or within-cluster sum-of-squares** . This algorithm **requires the number of clusters to be specified.** It scales well to large number of samples and has been used across a large range of application areas in many different fields.
- The k-means algorithm **divides a set of N samples X into K disjoint clusters C,** each described by the **mean μj** of the samples in the cluster. The means are commonly called the cluster **"centroids"; note that they are not, in general, points from X, although they live in the same space.**
- The K-means algorithm aims to choose centroids that minimise the **inertia**, or **within-cluster sum-of-squares criterion**:

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (||x_i - \mu_j||^2)$$

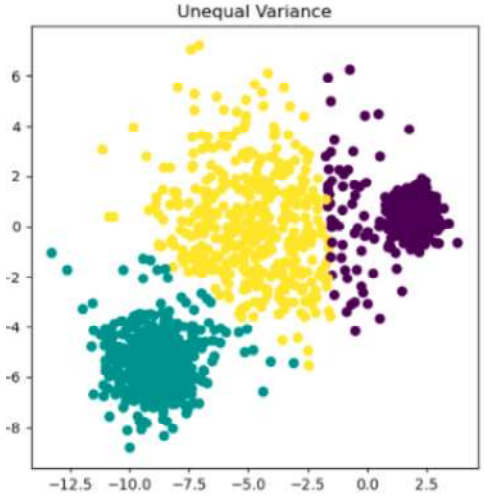
- **Inertia can be recognized as a measure of how internally coherent clusters are.** It suffers from various drawbacks:

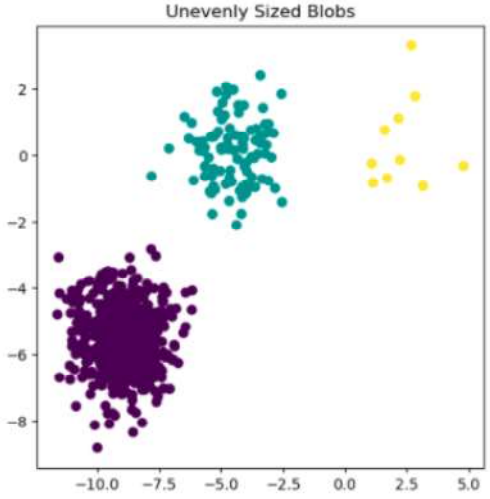
1. Inertia makes the **assumption** that **clusters are convex and isotropic**, which is not always the case. It **responds poorly to elongated clusters, or manifolds with irregular shapes**.
2. **Inertia** is not a normalized metric: we just know **that lower values are better and zero is optimal.** But in very **high-dimensional spaces, Euclidean distances tend to become inflated** (this is an instance of the so-called "curse of dimensionality"). Running a dimensionality reduction algorithm such as Principal component analysis (PCA) prior to k-means clustering can alleviate this problem and speed up the computations.





greatlearning





DBSCAN

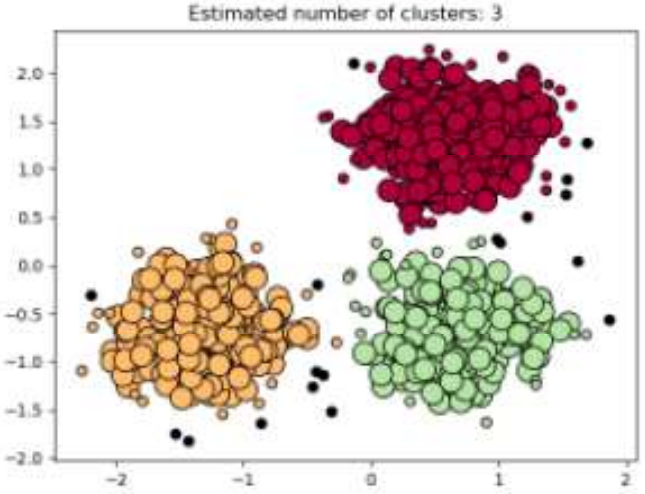
greatlearning

- The **DBSCAN** algorithm **views clusters as areas of high density separated by areas of low density**. Due to this rather generic view, **clusters found by DBSCAN can be any shape**, as opposed to k-means which assumes that clusters are convex shaped.
- The **central component** to the DBSCAN is the concept of ***core samples***, which are **samples that are in areas of high density**. A cluster is therefore a set of core samples, each close to each other and a set of non-core samples that are close to a core sample (but are not themselves core samples). There **are two parameters to the algorithm, min_samples and eps, which define formally what we mean when we say *dense*.**
  Higher min_samples or lower eps indicate higher density necessary to form a cluster.
- The parameter **eps** defines the radius of neighborhood around a point x. It's called called the ϵ-neighborhood of x
- **Minimum samples** ("MinPts"): the fewest number of points required to form a cluster. ε (epsilon or "eps"): the maximum distance two points can be from one another while still belonging to the same cluster.
- In other words, it defines **a core sample** as being a sample in the dataset such that **there exist min_samples other samples within a distance of eps**, which are defined as ***neighbors*** of the core sample. This tells us that the core sample is in a dense area of the vector space**. A cluster is a set of core samples that can be built by recursively taking a core sample**, finding all of its neighbors that are core samples, finding all of *their* neighbors that are core samples, and so on. **A cluster also has a set of non-core samples, which are samples that are neighbors of a core sample in the cluster but are not themselves core samples. Intuitively, these samples are on the fringes of a cluster.**
- Any core sample is part of a cluster, by definition. Any sample that is not a core sample, and is at least eps in distance from any core sample, is considered an **outlier** by the algorithm.

DBSCAN

greatlearning

- While the parameter **min_samples primarily controls how tolerant the algorithm is towards noise** (on noisy and large data sets it may be desirable to increase this parameter), the parameter **eps is *crucial to choose appropriately* for the data set and distance function** and usually cannot be left at the default value**. It controls the local neighborhood of the points. When chosen too small, most data will not be clustered at all (and labeled as -1 for "noise"). When chosen too large, it causes close clusters to be merged into one cluster, and eventually the entire data set to be returned as a single cluster.**
- In the figure below, the color indicates cluster membership, with large circles indicating core samples found by the algorithm. Smaller circles are non-core samples that are still part of a cluster. Moreover, the outliers are indicated by black points



MeanShift

greatlearning

- **MeanShift** clustering aims to discover *blobs* **in a smooth density of samples.** It is **a centroid based algorithm,** which works by **updating candidates for centroids to be the mean of the points within a given region.** These candidates are then filtered in a post-processing stage to eliminate near-duplicates to form the final set of centroids.
- Given a candidate centroid xi for iteration t, the candidate is updated according to the following equation:

$$x_i^{t+1} = m(x_i^t)$$

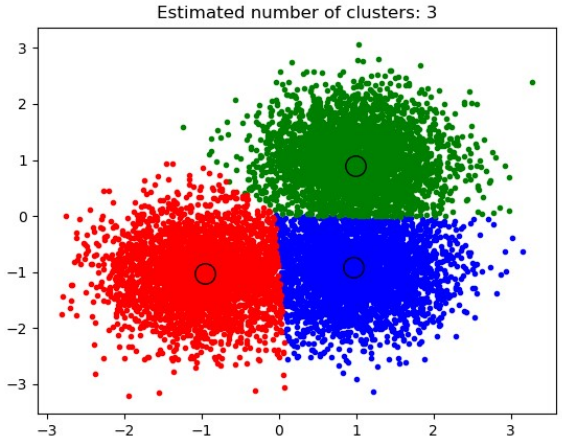
- Where N(xi) is the neighborhood of samples within a given distance around xi and m is the *mean shift* vector that is computed for each centroid that points towards a region of the maximum increase in the density of points. This is computed using the following equation, **effectively updating a centroid to be the mean of the samples within its neighborhood:**

$$m(x_i) = \frac{\sum_{x_j \in N(x_i)} K(x_j - x_i) x_j}{\sum_{x_j \in N(x_i)} K(x_j - x_i)}$$

MeanShift

greatlearning

- The algorithm **automatically sets the number of clusters**, instead of relying on a parameter bandwidth, which dictates the size of the region to search through. **This parameter can be set manually**, but can be estimated using the provided estimate_bandwidth function, which is called if the bandwidth is not set.
- The algorithm is **not highly scalable**, as it **requires multiple nearest neighbor searches during the execution of the algorithm**. The **algorithm is guaranteed to converge, however the algorithm will stop iterating when the change in centroids is small**.
- Labelling a new sample is performed by finding the nearest centroid for a given sample.



Hierarchical clustering

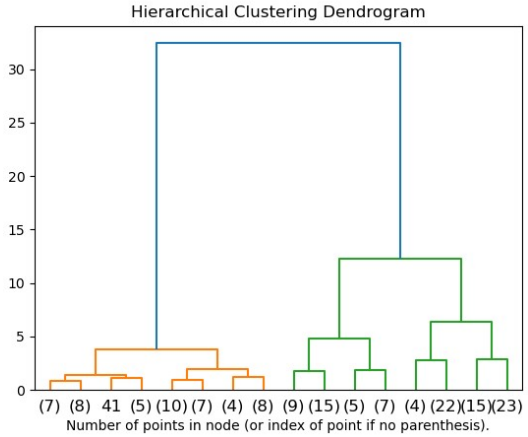
greatlearning

- **Hierarchical clustering** is a general family of clustering algorithms that build nested clusters by merging or splitting them successively. This hierarchy of clusters is represented as a tree (or dendrogram). The root of the tree is the unique cluster that gathers all the samples, the leaves being the clusters with only one sample.
- The **Agglomerative Clustering** object performs a hierarchical clustering using a bottom up approach: each observation starts in its own cluster, and clusters are successively merged together. The linkage criteria determines the metric used for the merge strategy:
- **Ward** minimizes the sum of squared differences within all clusters. It is a variance-minimizing approach and in this sense is similar to the k-means objective function but tackled with an agglomerative hierarchical approach.
- **Maximum** or **complete linkage** minimizes the maximum distance between observations of pairs of clusters.
- **Average linkage** minimizes the average of the distances between all observations of pairs of clusters.
- **Single linkage** minimizes the distance between the closest observations of pairs of clusters.
- **AgglomerativeClustering** can also scale to large number of samples when it is used jointly with a connectivity matrix, but is computationally expensive when no connectivity constraints are added between samples: it considers at each step all the possible merges.

Agglomerative Clustering

greatlearning

- Agglomerative cluster has a **“rich get richer” behavior that leads to uneven cluster sizes**. In this regard, **single linkage is the worst strategy, and Ward gives the most regular sizes**. However, the **affinity (or distance used in clustering) cannot be varied with Ward, thus for non Euclidean metrics, average linkage is a good alternative**. **Single linkage**, while **not robust to noisy data**, can be computed very efficiently and can therefore be useful to provide hierarchical clustering of larger datasets.
- It’s possible to visualize the tree representing the hierarchical merging of clusters as a dendrogram. Visual inspection can often be useful for understanding the structure of the data, though more so in the case of small sample sizes.



Clustering performance evaluation

greatlearning

- Evaluating the performance of a clustering algorithm is not as trivial as counting the number of errors or the precision and recall of a supervised classification algorithm. In particular any **evaluation metric should not take the absolute values of the cluster labels into account but rather if this clustering define separations of the data similar to some ground truth set of classes or satisfying some assumption such that members belong to the same class are more similar than members of different classes according to some similarity metric.**

1. **Rand index**
2. **Mutual Information based scores**
3. **Homogeneity, completeness and V-measure**
4. **Fowlkes-Mallows scores**
5. **Silhouette Coefficient**
6. **Calinski-Harabasz Index**
7. **Davies-Bouldin Index**
8. **Contingency Matrix**
9. **Pair Confusion Matrix**